State Complexity of Permutation on Finite Languages over a Binary Alphabet[☆]

Da-Jung Cho[a], Daniel Goč[b], Yo-Sub Han[a], Sang-Ki Ko[c], Alexandros Palioudakis[a,b], Kai Salomaa[b,*]

[a] *Department of Computer Science, Yonsei University, 50, Yonsei-Ro, Seodaemum-Gu, Seoul 120–749, Republic of Korea*
[b] *School of Computing, Queen's University, Kingston, Ontario K7L 3N6, Canada*
[c] *Department of Computer Science, University of Liverpool, Liverpool L69 3BX, United Kingdom*

Abstract

The set of all strings Parikh equivalent to a string in a language L is called the permutation of L. The permutation of a finite n-state DFA (deterministic finite automaton) language over a binary alphabet can be recognized by a DFA with $\frac{n^2-n+2}{2}$ states. We show that if the language consists of equal length binary strings the bound can be improved to $f(n) = \frac{n^2+n+1}{3}$ and for every n congruent to 1 modulo 3 there exists an n-state DFA A recognizing a set of equal length strings such that the minimal DFA for the permutation of $L(A)$ needs $f(n)$ states.

Keywords: state complexity, finite automata, finite languages, Parikh equivalence

1. Introduction

The descriptional complexity of finite automata is an active area of research. Recent surveys on descriptional complexity and state complexity include [10, 12, 19, 21]. The *state complexity of a regular language* L, $\mathrm{sc}(L)$, [30] is the minimal number of states of a deterministic finite automaton (DFA) recognizing the language. The number of states of the minimal DFA is equal to the number of left quotients of the language and the notion is equivalently called also *quotient complexity* [1]. The nondeterministic state complexity [18] measures the number of states of a minimal nondeterministic finite automaton (NFA) and transition

[☆] A preliminary version appeared in the *Proceedings of the 17th International Workshop on Descriptional Complexity of Formal Systems*, DCFS 2015, Lect. Notes Comput. Sci. 9118, Springer-Verlag, 2015, pp. 220–230.

[*] Corresponding author.

Email addresses: dajungcho@yonsei.ac.kr (Da-Jung Cho), goc@cs.queensu.ca (Daniel Goč), emmous@yonsei.ac.kr (Yo-Sub Han), sangkiko@liverpool.ac.uk (Sang-Ki Ko), alex@cs.queensu.ca (Alexandros Palioudakis), ksalomaa@cs.queensu.ca (Kai Salomaa)

complexity quantifies the number of transitions of an NFA [7, 11]. Also, the size of the syntactic semigroup of a minimal DFA has been used as a descriptional complexity measure for regular languages [1, 20, 15].

The effect of a regularity preserving operation on the number of states of the minimal DFA is called the *state complexity of the operation.* More precisely, if φ is an m-ary language operation, the state complexity of φ is a function $\varphi_{\rm sc} : \mathbb{N}^m \to \mathbb{N}$ such that for any regular languages $L_1, \ldots, L_m$, the language $\varphi(L_1, \ldots, L_m)$ has a DFA with $\varphi_{\rm sc}({\rm sc}(L_1), \ldots, {\rm sc}(L_m))$ states and, furthermore, for any $n_1, \ldots, n_m \in \mathbb{N}$ there exist regular languages L_i with ${\rm sc}(L_i) = n_i$, $i = 1, \ldots, m$, such that the minimal DFA for $\varphi(L_1, \ldots, L_m)$ has exactly $\varphi_{\rm sc}(n_1, \ldots, n_m)$ states [9].

The operational state complexity of regular languages was considered by Maslov [24] already in the 1970's. However, Maslov [24] gave no proofs and the work remained unknown in the west until much later. In 1994 Yu et al. [31] established the state complexity of basic regularity preserving operations and included complete proofs. Also the operational state complexity of subclasses of regular languages [8, 13], the nondeterministic state complexity of operations [16, 18], and the operational state complexity of NFAs with limited nondeterminism [25] have been considered in the literature. The precise worst-case state complexity of all combinations of two basic operations was determined by S. Yu and co-authors in a sequence of papers culminating with [5] and, interestingsly, it was shown by A. Salomaa et al. [26] that determining the state complexity of arbitrary combinations of operations that include marked concatenation and intersection is undecidable. Furthermore, Dassow and Harbich [6] have investigated the descriptional complexity, with respect to the number of productions and symbols, of operations on context-free languages. Operational state complexity is an active research topic and for more information we refer the reader to the survey [9].

The set of permutations of a string w consists of all strings w' such that the Parikh vectors of w and w' coincide. The permutation operation is extended in the natural way for languages and it is easy to see that the permutation operation does not, in general, preserve regularity. For example, the set of permutations of $(ab)^*$ is nonregular. We mention that Lavado et al. [23] have studied operational state complexity under Parikh equivalence, see also [22]. This work deals with finding the smallest DFA (or NFA) that is Parikh equivalent to a given language, as opposed to finding a DFA that recognizes all permutations of strings in a given language.

Here we investigate the state complexity of the permutation operation on finite languages. The state complexity of operations restricted to finite languages often is different from the general state complexity of the same operation [3, 14, 17]. Similarly as when considering the size blow-up of determinization of NFAs recognizing finite languages [27], the precise worst-case state complexity of permutation on finite languages depends on the alphabet size and we focus on binary alphabets.

We define state complexity as the minimal size of an incomplete DFA recognizing a language as this simplifies some of the constructions. Since we are

dealing with finite languages, as a corollary we get an exact bound also for the state complexity of permutation defined in terms of complete DFAs. Note that, in general, the relationship between the operational state complexity functions defined, respectively, in terms of complete and incomplete DFAs may be less clear. For example, for the shuffle operation the state complexity in terms of incomplete DFAs is known since 2002 [4], but determining the precise state complexity in terms of complete DFAs is more difficult and remains open [2].

First we give a general upper bound for the state complexity of the permutation of finite languages over binary alphabets. A chain DFA consists of a chain of transitions and only one final state (a more precise definition is given in section 2). We give an improved upper bound $f(n) = \frac{n^2+n+1}{3}$ for the state complexity of the permutation of an n state chain DFA and a matching lower bound. Chain DFAs recognize a subclass of equal length languages. We show that $f(n)$ is the precise worst case state complexity of the permutation of $L(A)$ where A is an n-state DFA recognizing an equal length language. The precise worst case state complexity of permutations of general finite languages remains open.

2. Preliminaries

We assume that the reader is familiar with the basic definitions concerning finite automata [28, 29] and just fix here some notation. General surveys on the descriptional complexity of finite automata include [9, 10, 19, 21].

The set of strings over a finite alphabet Σ is Σ^*, the length of $w \in \Sigma^*$ is $|w|$ and ε is the empty string. For a letter $a \in \Sigma$ and a string $w \in \Sigma^*$, we denote the number of occurrences of a in the string w by $|w|_a$. The set of positive integers is denoted by $\mathbb{N}$. The cardinality of a finite set S is $|S|$.

A deterministic finite automaton (DFA) is a 5-tuple $A = (Q, \Sigma, \delta, q_0, F)$, where Q is a finite set of states, Σ is a finite alphabet, δ is a partial function $Q \times \Sigma \to Q$, $q_0 \in Q$ is the start state and $F \subseteq Q$ is the set of accepting states. The function δ is extended in the usual way as a (partial) function $Q \times \Sigma^* \to Q$ and the language recognized by A consists of strings $w \in \Sigma^*$ such that $\delta(q_0, w) \in F$. The DFA A is *complete* if δ is a total function. When speaking of a DFA, unless otherwise mentioned, we allow the possibility that some transitions are undefined. By the *size of* A, size(A), we mean the number of states of A. Unless otherwise mentioned, we always assume that a DFA has no useless states, that is, each state can be reached from the initial state and a computation originating from each state can reach a final state.

We deal only with finite languages, so all DFAs we consider will be acyclic. We say that A is a *chain DFA* if the states of A can be numbered as $0, \ldots, n-1$, where 0 is the start state, $n-1$ is the only final state, and A has only transitions that take state i to $i+1$, $0 \leq i \leq n-2$. Note that for a state i transitions on some alphabet symbols can be undefined. A chain DFA with n states recognizes a subset of Σ^{n-1}.

Another subclass of acyclic DFAs we consider are the DFAs recognizing sets of equal length strings, called *equal length DFAs*. Note that every chain DFA is

an equal length DFA, but not vice versa. It is immediate that, for example, the set of equal length strings $\{aa, bb\}$ cannot be recognized by a chain DFA.

Lemma 2.1. *Let $L \subseteq \Sigma^\ell$, $\ell \in \mathbb{N}$. Then the minimal DFA for L has only one final state.*

Proof. Let A be the minimal DFA for L. Since A has no useless states and all strings accepted by A must have length ℓ, there can be no outgoing transitions from a final state. This means that all final states of A can be identified. □

The minimal size of a DFA recognizing a regular language L is called the state complexity of L and denoted by $\mathrm{sc}(L)$. Note that we allow DFAs to be incomplete. The convention simplifies some constructions since then we do not need to include a dead state and large numbers of transitions to the dead state. The minimal incomplete DFA for a finite language L has always exactly one less state than the minimal complete DFA for L and, since all our state complexity bounds deal with finite languages, the precise state complexity bounds can be straightforwardly translated for upper and lower bounds defined in terms of complete DFAs (see Corollary 4.4).

The (right) Myhill-Nerode congruence of a language $L \subseteq \Sigma^*$ is the relation $\equiv_L \subseteq \Sigma^* \times \Sigma^*$ defined by setting

$$x \equiv_L y \text{ iff } [(\forall z \in \Sigma^*)\ \ xz \in L \Leftrightarrow yz \in L].$$

The language L is regular if and only if the index of $\equiv_L$ is finite and, in this case, the index of $\equiv_L$ is equal to the size of the minimal complete DFA for L [28].

The permutation $\mathrm{per}(L)$ of a language $L \subseteq \Sigma^*$ consists of all strings $w \in \Sigma^*$ such that for some string $u \in L$ the strings w and u have the same number of occurrences of every letter of Σ. Formally, we define

$$\mathrm{per}(L) = \{\ w \in \Sigma^* \mid (\exists u \in L)(\forall a \in \Sigma)(|u|_a = |w|_a)\ \}.$$

Note that the family of regular languages is not closed under permutation. For example, the permutation of $(a{\cdot}b)^*$ consists of all strings w such that $|w|_a = |w|_b$, which is not a regular language. On the other hand, if L is finite so is $\mathrm{per}(L)$.

Given an alphabet $\Sigma = \{a_1, a_2, \ldots, a_k\}$, let $\Psi : \Sigma^* \to [\mathbb{N}_0]^k$ be a mapping defined by $\Psi(w) = (|w|_{a_1}, |w|_{a_2}, \ldots, |w|_{a_k})$. This function is called the *Parikh mapping* and $\Psi(w)$ is called the *Parikh vector* of w. The Parikh mapping is extended for sets of strings, $\Psi : 2^{\Sigma^*} \to 2^{[\mathbb{N}_0]^k}$, in the natural way by setting $\Psi(L) = \{\Psi(w) \mid w \in L\}$. Two languages L_1, L_2 are Parikh equivalent, denoted by $L_1 \equiv_{\mathrm{Parikh}} L_2$, if $\Psi(L_1) = \Psi(L_2)$.

3. General upper bound for the state complexity of permutation

We begin by giving an upper bound for the state complexity of permutation for arbitrary finite languages over a binary alphabet. In the next section we derive a tight state complexity bound for chain DFAs.

In the following, unless otherwise mentioned, we always assume that the alphabet is $\Sigma = \{a, b\}$.

Lemma 3.1. *Let $L \subseteq \{a, b\}^*$ be a finite language and $m = \max\{|w| \mid w \in L\}$, for some positive integer m. Then*

$$\mathrm{sc}(\mathrm{per}(L)) \leq \frac{m^2 + m + 2}{2}.$$

Proof. We construct a DFA $A = (Q, \{a, b\}, \delta, q_0, F)$ that recognizes $\mathrm{per}(L)$. The set of states of A will be

$$Q = \{\ (i, j) \mid 0 \leq i, j < m \text{ and } i + j < m\ \} \cup \{f_A\},$$

the set of final states is $F = \{f_A\} \cup \{\ (i, j) \in Q \mid a^i b^j \in \mathrm{per}(L)\ \}$, and the partial transition function δ is defined by setting for $(i, j) \in Q$,

$$\delta((i,j), a) = \begin{cases} (i+1, j) & \text{if } i + j + 1 < m; \\ f_A & \text{if } i + j + 1 = m \text{ and } a^{i+1}b^j \in \mathrm{per}(L); \\ \text{undefined} & \text{if } i + j + 1 = m \text{ and } a^{i+1}b^j \notin \mathrm{per}(L). \end{cases}$$

$$\delta((i,j), b) = \begin{cases} (i, j+1) & \text{if } i + j + 1 < m; \\ f_A & \text{if } i + j + 1 = m \text{ and } a^{i}b^{j+1} \in \mathrm{per}(L); \\ \text{undefined} & \text{if } i + j + 1 = m \text{ and } a^{i}b^{j+1} \notin \mathrm{per}(L). \end{cases}$$

The transitions $\delta(f_A, a)$ and $\delta(f_A, b)$ are undefined.

A state of A keeps track of the number of occurrences of a's and of b's that have been read in the input so far. Thus, for all states (i, j) of A we can restrict $i + j$ to be at most $m - 1$ and f_A is used as an accepting state when the count has reached $i + j = m$. The final states of A are all states (i, j) such that there is a string $w \in L$ with $|w|_a = i$ and $|w|_b = j$.

The number of states of A is $m + m - 1 + \cdots + 1 + 1 = \frac{m \cdot (m+1)}{2} + 1$. □

For a finite language L, the length of the longest string of L is at most $\mathrm{sc}(L) - 1$. By this observation and Lemma 3.1, we have the following corollary.

Corollary 3.2. *Let L be a binary finite language and $\mathrm{sc}(L) = n$ for some positive integer n. Then*

$$\mathrm{sc}(\mathrm{per}(L)) \leq \frac{n^2 - n + 2}{2}.$$

The upper bound of Corollary 3.2 is not optimal and, for large values of n, could be improved using a more careful analysis of the construction of Lemma 3.1.[1] The precise state complexity of permutation of finite languages remains open. In the next sections we derive tight bounds for languages recognized by chain DFAs and, more generally, for sets of equal length strings.

[1] This has been pointed out by a referee of the paper.

4. State complexity of permutation on chain DFAs

We show that for languages recognized by chain DFAs, in the upper bound of Corollary 3.2, roughly speaking, the multiplicative constant $\frac{1}{2}$ can be reduced to $\frac{1}{3}$ and that, furthermore, the resulting bound is tight.

Lemma 4.1. *Let A be a chain DFA with n states over the alphabet $\{a, b\}$. Then*

$$\mathrm{sc}(\mathrm{per}(L(A))) \leq \frac{n^2 + n + 1}{3}.$$

Proof. Let the set of states of A be $\{1, \ldots, n\}$, where 1 is the start state and n is the only final state, and for $1 \leq h \leq n-1$, the transitions from h go only to $h+1$. We have three possibilities for outgoing transitions from a state $1 \leq h \leq n-1$:

1. $\delta(h, a) = h + 1$ and $\delta(h, b)$ is undefined (a-transition)
2. $\delta(h, b) = h + 1$ and $\delta(h, a)$ is undefined (b-transition)
3. $\delta(h, a) = \delta(h, b) = h + 1$ ($a\&b$-transition)

The order of the different types of transitions (a, b, or $a\&b$) of A does not affect the language $\mathrm{per}(L(A))$. Hence, without loss of generality, we can assume that the DFA A has first a (possibly empty) sequence of a-transitions, followed by a (possibly empty) sequence of b-transitions, followed by a (possibly empty) sequence of $a\&b$-transitions. Thus, we can assume that A recognizes a language of the form $a^i b^j (a+b)^k$ for some non-negative integers i, j, k such that $i + j + k = n - 1$.

Now the language $\mathrm{per}(L(A))$ is recognized by the DFA $B_{i,j,k} = (Q, \{a, b\}, \gamma, q_0, F_B)$ where

$$\begin{aligned} Q &= \{\ (r, s)\ \mid\ 0 \leq r \leq i + k,\ 0 \leq s < j\ \} \\ &\cup \{\ (r, s)\ \mid\ 0 \leq s \leq j + k,\ 0 \leq r < i\ \} \cup \{z_0, z_1, \ldots, z_k\}, \end{aligned}$$

$F_B = \{z_k\}$, $q_0 = (0, 0)$ and the transitions are defined by setting, for $(r, s) \in Q$,

$$\gamma((r, s), a) = \begin{cases} (r + 1, s) & \text{if } r < i - 2 \text{ or } (r < i + k \text{ and } s < j); \\ z_{s-j} & \text{if } r = i - 1 \text{ and } s \geq j; \\ \text{undefined} & \text{if } r + 1 > i + k; \end{cases}$$

$$\gamma((r, s), b) = \begin{cases} (r, s + 1) & \text{if } s < j - 2 \text{ or } (s < j + k \text{ and } r < i) \\ z_{r-i} & \text{if } s = j - 1 \text{ and } r \geq i; \\ \text{undefined} & \text{if } s + 1 > j + k; \end{cases}$$

and,

$$\gamma(z_\ell, a) = \gamma(z_\ell, b) = \begin{cases} z_{\ell+1} & \text{if } 0 \leq \ell < k, \\ \text{undefined} & \text{if } \ell = k. \end{cases}$$

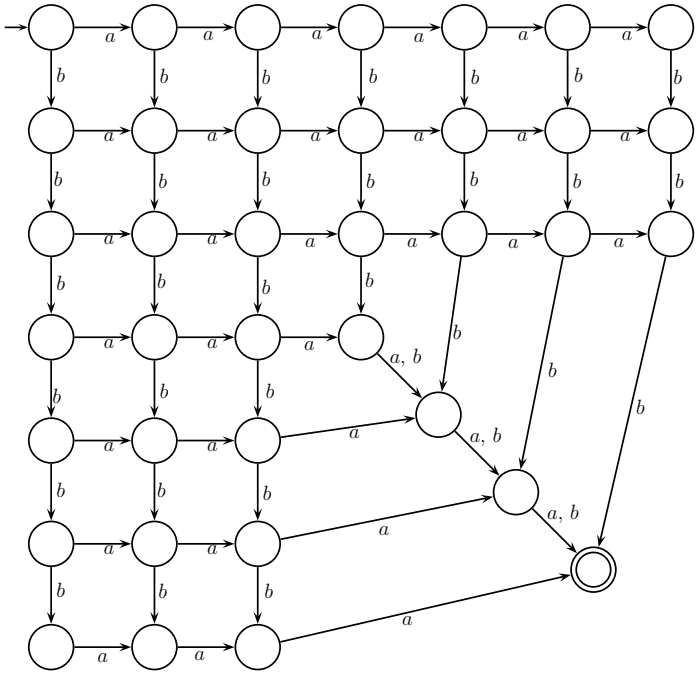

Figure 1: The DFA $B_{3,3,3}$.

The DFA $B_{3,3,3}$ is given in Figure 1. A computation of $B_{i,j,k}$ reaches a state of the form (r, s) after encountering r occurrences of a and s occurrences of b, where $r < i$ or $s < j$. A state z_ℓ, $0 \leq \ell \leq k$, is reached after encountering at least i occurrences of a and at least j occurrences of b, where $\ell + i + j$ is the length of the input processed up to that point. Thus, $B_{i,j,k}$ reaches the accepting state z_k exactly on inputs of length $i + j + k$ that have at least i occurrences of a and at least j occurrences of b.

The cardinality of Q is

$$f(i, j, k) = (i + 1) \cdot (j + 1) + k \cdot j + k \cdot i + k$$

(taking into account that the first two sets in the union defining Q have elements in common). In order to get an upper bound for the state complexity of $\mathrm{per}(L)$ as a function of the size of A, we determine for which values of i, j, k, where $i + j + k = n - 1$, the function $f(i, j, k)$ has a maximal value. The function f is maximized if $ij + kj + ki$ is maximal, thus if $i = j = k = \frac{n-1}{3}$. More generally,

$$\max_{i+j+k=n-1} f(i, j, k) = \begin{cases} \frac{n^2+n+1}{3}, & \text{if } n \equiv 1 \pmod 3; \\ \frac{n^2+n}{3}, & \text{otherwise.} \end{cases}$$

□

Next we show that the upper bound of Lemma 4.1 can be reached by a chain DFA language.

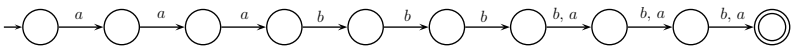

Figure 2: A chain DFA recognizing the language L_{10}.

Lemma 4.2. *For any $n \in \mathbb{N}$ with $n \equiv 1 \pmod 3$ there exists a binary regular language recognized by a chain DFA with n states such that*

$$\mathrm{sc}(\mathrm{per}(L)) \geq \frac{n^2+n+1}{3}.$$

Proof. Denote $n = 3k+1$, $k \geq 0$ and choose $L_n = a^k b^k (a+b)^k$. For $k = 3$ and $n = 10$, Figure 2 gives a chain DFA with 10 states recognizing L_{10}.

We prove a lower bound for the state complexity of $\mathrm{per}(L_n)$. Note that with $n = 3k+1$ we have

$$\mathrm{per}(L_n) = \{\ w \in \Sigma^{3\cdot k} \ \mid\ |w|_a, |w|_b \geq k\ \}.$$

Let X and Y be the sets of strings chosen as follows:

$$X = \{a^i b^j : 0 \leq i \leq 2k, 0 \leq j \leq k\} \text{ and } Y = \{a^i b^j : 0 \leq i < k, k < j \leq 2k\}.$$

We show that all strings of $X \cup Y \cup \{a^{2k+1}\}$ are pairwise inequivalent with respect to the Myhill-Nerode congruence of $\mathrm{per}(L_n)$. Every string of $X \cup Y$ is a prefix of a string in $\mathrm{per}(L_n)$ and this implies that a^{2k+1} is not equivalent with any string of $X \cup Y$. It remains to show that all strings of $X \cup Y$ are pairwise inequivalent. Let $u = a^i b^j$ and $u' = a^{i'} b^{j'}$ be two arbitrary distinct strings from $X \cup Y$.

First consider the possibility that $|u| \neq |u'|$. Now u and u' are inequivalent since by choosing a string z such that $uz \in \mathrm{per}(L_n)$ we note that $|u'z| \neq 3 \cdot k$ and, consequently, $u'z \notin \mathrm{per}(L_n)$. Thus, in the following case analysis we can assume that $|u| = |u'|$.

1. Consider the possibility $u, u' \in X$. Since $|u| = |u'|$ and $u \neq u'$, either $|u|_a < |u'|_a$ or $|u'|_a < |u|_a$. Without loss of generality, we assume that $|u|_a < |u'|_a$. For $z = a^{2\cdot k - i} b^{k-j}$, we have $uz \in \mathrm{per}(L_n)$. However, for the string $u'z$, we have $|u'z|_a > 2 \cdot k$, which means, since $|uz| = |u'z| = 3 \cdot k$, that $|u'z|_b < k$ and $u'z \notin \mathrm{per}(L_n)$.
2. Next consider the case $u, u' \in Y$. Similarly as above, without loss of generality, we assume that $|u|_b < |u'|_b$. For $z = a^{k-i} b^{2\cdot k - j}$, we have $uz \in \mathrm{per}(L_n)$. However, we have $|u'z|_b > 2 \cdot k$, which implies that $u'z \notin \mathrm{per}(L_n)$.
3. The remaining possibility is that $u \in X$ and $u' \in Y$. Since $u' \in Y$ and $u \in X$, we know that $|u'|_b > k$ and $|u|_b \leq k$. This implies that $|u|_a > |u'|_a$ because $|u| = |u'|$.
   (a) First consider the case $i > k$ and choose $z = b^{3k-i-j}$. Now $uz \in \mathrm{per}(L)$ but $|u'z|_a < k$ and, consequently, $u'z \notin \mathrm{per}(L)$.
   (b) Second consider the case $i \leq k$. Now for the string $z = a^{k-i} b^{2\cdot k - j}$, we have $uz \in \mathrm{per}(L_n)$. However, for the string $u'z$, we have that $|u'z|_a < k$ and, thus, $u'z \notin \mathrm{per}(L_n)$.

For finite languages, the number of states of the minimal incomplete DFA is one less than the index of the Myhill-Nerode congruence and, hence, the number of states of the minimal incomplete DFA recognizing the language $\mathrm{per}(L_n)$ has at least $(2 \cdot k + 1) \cdot (k + 1) + k^2 = 3 \cdot k^2 + 3 \cdot k + 1$ states. Since $n = 3 \cdot k + 1$, a simple calculation yields $\mathrm{sc}(\mathrm{per}(L_n)) \geq \frac{n^2+n+1}{3}$. □

From Lemma 4.1 it follows that the lower bound given in Lemma 4.2 (for values $n \equiv 1 (\mathrm{mod}\ 3)$) is also an upper bound for the state complexity of $\mathrm{per}(L_n)$, $n \in \mathbb{N}$. The minimal DFA recognizing the language $\mathrm{per}(L_{10})$ is shown in Figure 1.

Now combining the results of Lemma 4.1 and Lemma 4.2 we have a tight bound for the state complexity of permutation for languages defined by chain DFAs.

Theorem 4.3. *Let $n \in \mathbb{N}$ and let L be a binary language recognized by a chain DFA with n states. Then*

$$\mathrm{sc}(\mathrm{per}(L)) \leq \frac{n^2 + n + 1}{3}.$$

For every $n \equiv 1$ (mod 3), there exists a binary chain DFA A with n states such that $\mathrm{sc}(\mathrm{per}(L(A))) = \frac{n^2+n+1}{3}$.

Recall that our state complexity definition is based on the size of incomplete DFAs. Since for a finite language the size of the minimal incomplete DFA is always exactly one less than the size of the minimal complete DFA, translating the upper bound of Lemma 4.1 and the lower bound of Lemma 4.2 for complete DFAs yields the same bound in both cases. By a *complete chain DFA* we mean a chain DFA A augmented with a dead state that is the target of all undefined transitions of A.

Corollary 4.4. *If L is recognized by a complete chain DFA with n states, the language $\mathrm{per}(L)$ is recognized by a complete DFA with $g(n) = \frac{n^2-n+4}{3}$ states.*

For every $n \equiv 2$ (mod 3), there exists a complete chain DFA A with n states such that the minimal complete DFA for $\mathrm{per}(L(A))$ needs $g(n)$ states.

5. Upper bound for sets of equal length strings

We prove an upper bound for the state complexity of permutation of sets of equal length strings. The upper bound coincides with the lower bound from Lemma 4.2, which uses chain DFAs that define a subclass of equal length languages. We begin by introducing some terminology for DFAs that recognize sets of equal length strings.

5.1. Terminology and notation for equal length DFAs

In the following, we consider a DFA $A = (Q, \{a, b\}, \delta, q_0, \{q_f\})$ recognizing a subset of $\{a, b\}^\ell$. Without loss of generality A has no useless states and by

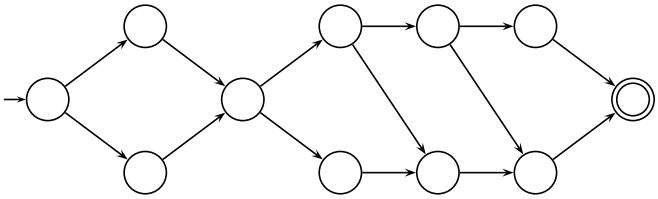

Figure 3: A DFA with a block of length 2 and a block of length 4.

Lemma 2.1 we can assume that A has one final state. The number of states of A is always n.

The *level of a state* $q \in Q$ is the length of a string w such that $\delta(q_0, w) = q$. Since A recognizes a set of equal length strings and has no useless states, the level of a state is a unique integer in $\{0, 1, \ldots, \ell\}$. The set of level z states is $Q[z]$ for $0 \leq z \leq \ell$. We say that level z is *singular* if $|Q[z]| = 1$, $0 \leq z \leq \ell$. Levels 0 and ℓ are always singular. A *linear transition* is a transition between two singular levels. A linear transition can be labeled by a, b or $a\&b$. (A linear transition labeled by $a\&b$ is strictly speaking two transitions.) The number of linear transitions labeled by a (respectively, by b, $a\&b$) is denoted i (respectively, j, k).

The length of the nonlinear part of A is

$$h = \ell - (i + j + k). \tag{1}$$

Thus h denotes the number of pairs $(z, z+1)$, for $0 \leq z < \ell$, such that at least one of the levels z or $z+1$ is not singular.

Consider $0 \leq x \leq \ell$, $0 \leq y \leq \ell$, and $x + 1 < y$, where levels x and y are singular and all levels strictly between x and y are non-singular. A *nonlinear block* $B_{x,y}$ of A between the levels x and y is a subautomaton of A consisting of all states of $\cup_{x \leq z \leq y} Q[z]$ and the transitions between them. The initial (respectively, final) state of the subautomaton is the state having level x (respectively, level y). The length of the nonlinear block $B_{x,y}$ is $y - x$. The length of a block is always at least two.

Note that a nonlinear block begins and ends in a singular level and all levels between these are non-singular. In the following, nonlinear blocks are called simply *blocks.* Examples of blocks are illustrated in Figure 3.

The sum of the lengths of the blocks of A equals to h. The estimation of the length of accepted strings ℓ in terms of the number of states n depends on the types of blocks that A has.

Assume that the total length h of the blocks of A is fixed. Then the maximal value of ℓ can be reached if all blocks have length two (and h is even). Note that a block of length two has always exactly 4 states. Thus, we have

$$\ell \leq n - 1 - \frac{1}{2}h. \tag{2}$$

Example of the worst-case situation where $\ell = n - 1 - \frac{1}{2}h$ is illustrated in Figure 4.

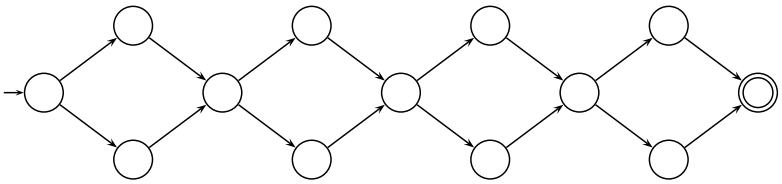

Figure 4: $n = 13$ and $h = 8$, $\ell = 8$.

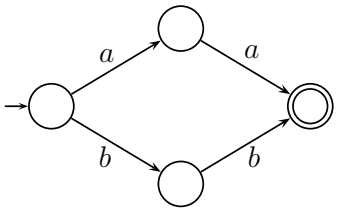

Figure 5: A diamond.

5.2. Estimate for DFAs having blocks of length two

We begin by providing an upper bound in the case where a DFA A includes blocks of length two and none of bigger length. Strictly speaking, this is a special case of the more general estimate given in Section 5.3. The proof of the general case is based on similar ideas but is considerably more complicated. For readability we consider first the special case of DFAs having blocks of length at most two.

A block of length two that recognizes the language $\{aa, bb\}$ is called a *diamond* (see Figure 5). There are a total of 9 different blocks of length two and it is easy to see that any block of length two that is not a diamond is "redundant" in the sense that it can be replaced by linear transitions and the modified DFA is Parikh-equivalent to A and has fewer states. This is stated in the following lemma.

Lemma 5.1. *Assume that A has a block of length two that is not a diamond. Then there exists a DFA A_1 having $n-1$ states such that $L(A_1) \equiv_{\text{Parikh}} L(A)$.*

In the next lemma, we observe that if A has one or more diamonds, then without loss of generality A can be assumed to have no linear transitions with label $a\&b$.

Lemma 5.2. *Assume that A has $r \geq 1$ diamonds and $k \geq 1$. Then there exists a DFA A_2 with $n-r$ states such that $L(A_2) \equiv_{\text{Parikh}} L(A)$.*

Proof. This follows from the observation that when $k \geq 1$,

$$(aa + bb)^r (a + b)^k \equiv_{\text{Parikh}} (a + b)^{2r+k}.$$

□

By Lemmas 5.1 and 5.2, when computing an upper bound estimate for the state complexity of per($L(A)$), in the case where A has blocks of length two, we can assume that all blocks of length two are all diamonds and, furthermore, that $k = 0$ (i.e., A has no linear transitions labeled with $a\&b$).

With the above assumptions combining with (1) and (2), we have

$$\frac{3}{2} \cdot h + i + j \leq n - 1. \tag{3}$$

We construct a DFA B recognizing per($L(A)$). Note that it is sufficient for B to count a's up to $i + h$ and count b's up to $j + h$ with the further restriction that the sum of the counts is at most $i + j + h$. The states of B consist of pairs (x, y), where x is the a-count and y is the b-count. The states can be listed as follows:

- $(i + 1) \cdot (j + 1)$ pairs, where a-count is at most i and b-count is at most j.
- When a-count is $i + z$, for $1 \leq z \leq h$, b-count can be between 0 and $j + h - z$. This results in $\frac{1}{2}h(2j + h + 1)$ states. (The number of states comes from calculating, for some positive integers m and n, the cardinality of the following set $\{(i_0, j_0) \mid 1 \leq i_0 \leq m, 0 \leq j_0 \leq n + m - i_0\}$ which is $\frac{1}{2}m(2n + m + 1)$.)
- Additionally, for each b-count greater than j, we need to count up to i a's, which results in $h \cdot (i + 1)$ added states. The situation where also the a-count is above i was included already in states listed above.

In total, B has

$$ij + hi + hj + \frac{1}{2}h^2 + i + j + \frac{3}{2}h + 1$$

states. This function, under constraint (3) can be maximized by mathematics software such as Maple. For the sake of completeness, we include here a proof.

Substituting $h = \frac{2}{3}(n - 1 - i - j)$ yields a two variable function that after some simplification can be written as

$$f(i, j) = ij + n + \frac{2}{9}(ni + nj - i - j - 2i^2 - 2j^2 - 4ij + n^2 - 2n + 1).$$

1. Boundary $j = 0$. The maximum is attained at $i = \frac{n-1}{4}$ and $f(\frac{n-1}{4}, 0) = \frac{n^2}{4} + \frac{n}{2} + \frac{1}{4}$ and this polynomial is always at most $\frac{n^2+n+1}{3}$.
2. The calculation for the boundary $i = 0$ is symmetric.
3. Critical points of $f(i, j)$: Setting the first partial derivative as zero, after some simplification we get

$$\frac{\partial f}{\partial i} = \frac{2}{9}(n - 1 - 4i + \frac{1}{2}j) = 0,$$

that is, $n-1 = 4i-\frac{1}{2}j$. Symmetrically, setting $\frac{\partial f}{\partial j} = 0$ yields $n-1 = 4j-\frac{1}{2}i$.

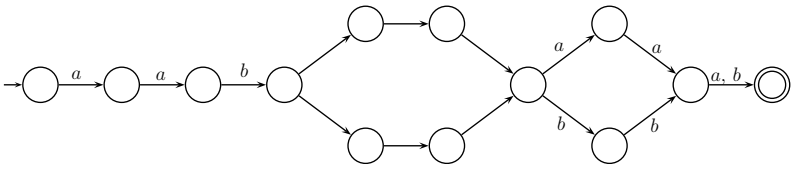

Figure 6: An NFA of the form specified in the proof of Theorem 5.4.

Solving the equations $\frac{\partial f}{\partial i} = 0$ and $\frac{\partial f}{\partial j} = 0$ gives the solution $i = j = \frac{2}{7}(n-1)$, and consequently $h = \frac{2}{7}(n-1)$. This gives a maximal value for $f(i,j)$ as $\frac{2}{7}(n-1)^2 + n$. This polynomial is upper bounded by $\frac{n^2+n+1}{3}$ and only reaches that bound in the trivial case where $i = j = h = 0$ and $n = 1$.[2]

Above we have verified the following:

Proposition 5.3. *If A is a DFA with n states that recognizes a set of equal length strings over $\{a, b\}$ and the nonlinear part of A has only blocks of length two, then*

$$\mathrm{sc}(\mathrm{per}(L(A)) \leq \frac{n^2+n+1}{3}.$$

5.3. Estimate for general DFAs for equal length languages

The result of the following theorem extends the result of Proposition 5.3 to all DFAs recognizing sets of equal length strings.

Theorem 5.4. *Let A be an n-state DFA recognizing a language $L \subseteq \{a, b\}^\ell$. Then there exists a DFA C recognizing* $\mathrm{per}(L)$ *with no more than $\frac{n^2+n+1}{3}$ states.*

Proof. Without loss of generality, we assume that A is of the following form: a set of linear transitions labelled by a, followed by a set of linear transitions labelled by b, followed by a blocks ordered in decreasing length and finally followed by linear transitions labelled by $a\&b$. An example of such an NFA can be seen in Figure 6. Any DFA that is not of this form can be 'rearranged' into a Parikh-equivalent DFA of this form. Furthermore, we assume that the DFA A is minimal in its Parikh-equivalence class, since only the minimal DFAs of a language will contribute to the bound.

Next, we define a function which measures the maximum distance between the two closest words in a language L. Let $L = \{w_1, w_2, \ldots, w_p\} \in \{a, b\}^\ell$ and order the w_i so that $|w_1|_a \leq |w_2|_a \leq \cdots \leq |w_p|_a$. Then define $\gamma(L) = \max_{1 \leq i \leq p-1}\{|w_{i+1}|_a - |w_i|_a\}$.

[2]Note that the general construction used in section 4 to reach this bound uses $a\&b$ transitions, i.e., $k > 0$. Our current estimate could exclude $a\&b$ transitions in Lemma 5.2 because in the presence of a block of length two the smallest Parikh equivalent DFA cannot have $a\&b$ transitions.

Now γ gives useful insight into the state complexity of $L \in \{a,b\}^\ell$ or at least the state complexity of another language Parikh-equivalent to L.

Claim 1. Suppose that $\gamma(L) = 1$, and s and t be respectively the lowest a-count and b-count of all the words in L. Then we claim that the language $K = a^s b^t (a+b)^{\ell-(s+t)}$ is Parikh-equivalent to L.

Proof of Claim 1. Let $w \in L$ and let $\psi(w) = (w_a, w_b)$. We know that $s \leq w_a \leq \ell - t$ since there must be at least s a's in w and at least t b's. Similarly, we conclude that $t \leq w_b \leq \ell - s$. Now, $v = a^s b^t a^{w_a - s} b^{w_b - t} \in K$ and $\psi(v) = (w_a, w_b)$, so we have that $\psi(L) \subseteq \psi(K)$.

On the other hand, let $v \in K$ and let $\psi(v) = (v_a, v_b)$. Again, we conclude that $s \leq v_a \leq \ell - t$ and $t \leq v_b \leq \ell - s$. Suppose that $(v_a, v_b) \notin \psi(L)$, and let $w_i \in L$ be such that $|w_i|_a < v_a < |w_{i+1}|_a$ (consequently $|w_i|_b > v_b > |w_{i+1}|_b$). Such a w_i must exist since $(s, \ell - s) \in \psi(L)$ and $(\ell - t, t) \in \psi(L)$. Then $|w_{i+1}|_a - |w_i|_a \geq 2$, a contradiction. This implies that $\psi(K) \subseteq \psi(L)$ concluding the proof of our claim. $\triangleleft$

Let m be the length of the shortest block in A. Thus, in particular, $m \leq h$. We have two cases to consider:

Case $m \leq k+1$**:** Let B the last nonlinear block of A (since the blocks of A are ordered in decreasing size of the blocks, B is of length m – the length of the shortest block) and let s and t be respectively the lowest a-count and b-count of all the words in $L(B)$. It follows that $\gamma(L(B))$ is at most m (in fact no more than $m - s - t$) but since $m \leq k+1$ we have that $\gamma(L(B) \cdot (a+b)^k)$ is 1.

Proof of Case 1. Suppose that $\gamma(L(B) \cdot (a+b)^k) > 1$ and write $L(B) \cdot (a+b)^k = \{w_1, w_2, \ldots, w_p\}$ where $|w_1|_a \leq |w_2|_a \leq \cdots \leq |w_p|_a$. Let i be such that $|w_i|_a < |w_{i+1}|_a - 1$. Suppose that there is a b in the last k letters of w_i, clearly it can be replaced with a forming $w' \in L(B) \cdot (a+b)^k$ a contradiction. Similarly we argue that the last k letters of w_{i+1} contain no a. Therefore, $w_i = w_i' a^k$ and $w_{i+1} = w_{i+1}' b^k$ where w_i' and w_{i+1}' belong to $L(B)$. But $\gamma(L(B)) \leq m \leq k+1$ and so $|w_{i+1}'|_a - |w_i'|_a \leq k+1$ which implies $|w_{i+1}|_a - |w_i|_a \leq -k+k+1 = 1$. This contradicts our assumption and so we conclude that $\gamma(L(B) \cdot (a+b)^k) = 1$. In Figure 7 we can see an example of a language K with $\gamma(K) = 4$ showing how $\gamma(K \cdot (a+b)^3) = 1$. $\triangleleft$

Let A' be the machine modified from A where the block B is replaced with a subautomaton with m linear transitions recognizing the language $a^s b^t (a+b)^{m-(s+t)}$. By Claim 1, A' is Parikh-equivalent to A, but clearly A' has less states than A as all the nonlinear states in B are replaced with one state per level. This contradicts A's minimality.

Case $m > k+1$**:** Next we calculate an estimate for DFAs that have no blocks of length less than $k+2$. In this case we get a better estimate for ℓ in terms of n and h than the general bound provided by (2). For example, when h is fixed, and $m = 3$ the maximal value of ℓ is reached when blocks of A are as depicted in Figure 8. (We have already illustrated the example

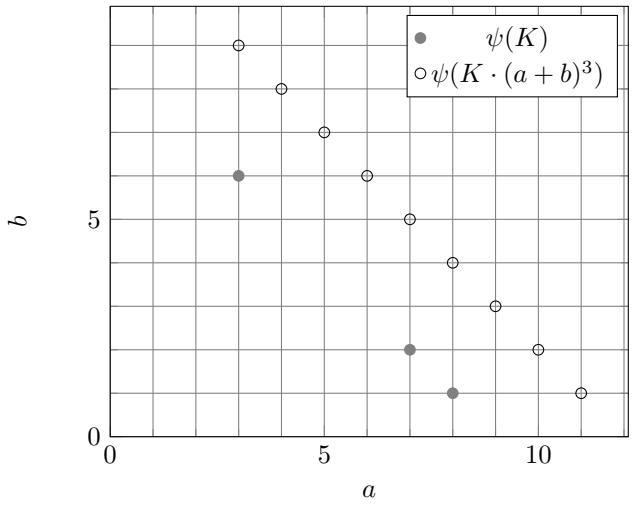

Figure 7: A figure showing $\psi(K \cdot (a+b)^3)$ where $\gamma(K) = 4$.

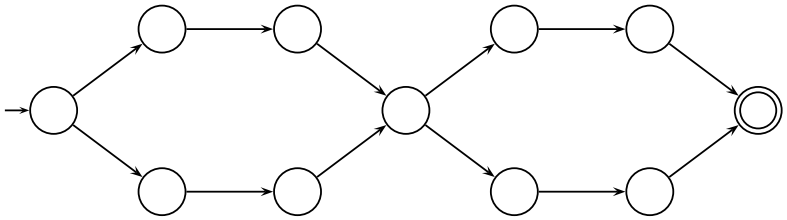

Figure 8: Two blocks of length 3.

when $m = 2$ in the Figure 4.) In other words, the maximal value of ℓ is reached when all the blocks are as short as possible – in this case m. Since for each block of length m we need at least $m - 1$ non-singular levels we get

$$\ell \leq n - 1 - \frac{m}{2m-1} \cdot h. \tag{4}$$

A DFA C recognizing $\mathrm{per}(L(A))$ counts a's up to $i + h$, b's up to $j + h$ and checks that the length of the string is $i + j + k + h$. The number of "counting" states can be reduced by the observation that the sum of the counts should be at most $i + j + k + h$. The number of states that can be reduced is expressed in the formula g further below. Having that $m \geq k + 2$ we see that g should be maximised under the condition:

$$\frac{2k+3}{k+2} \cdot h + i + j + k \leq n - 1. \tag{5}$$

In this case for the DFA C it is sufficient to have following states (x, y) where x is the a-count and y is the b-count.

- $0 \le x \le i+k$, $0 \le y \le j+k$ which gives $(i+1+k)\cdot(j+1+k)$ states,
- – when a-count $x = i+k+1$, count b's in range $0 \le y \le j+h-1$ This yields $j+h$ states.
  – when $x = i+k+2$, need $j+h-1$ states,
  – $\ldots$,
  – when $x = i+h+k$, need $j+1$ states.

  In total this part uses $\frac{1}{2}\cdot h\cdot(2j+h+1) = hj+\frac{1}{2}h^2+\frac{1}{2}h$ states.
- – when b-count is $y = j+k+1$, count a's between $0 \le x \le i+k$. (The values $x > i+k$ were included in the previous item.)
  – $\ldots$
  – when b-count is $y = j+h$, count a's between $0 \le x \le i+k$.

  This part yields $(h-k)\cdot(i+k+1)$ states.
- – when b-count $y = j+h+1$, count $0 \le x \le i+k-1$.
  – $\ldots$,
  – when $y = j+h+k$, count $0 \le x \le i$.

  This part yields $\frac{1}{2}\cdot k\cdot(2i+k+1) = ki+\frac{1}{2}\cdot k^2+\frac{1}{2}k$ states.

In total the above construction gives

$$g(i,j,k,h) = ij+ik+jk+hi+hk+hj+\frac{1}{2}h^2+\frac{1}{2}k^2+i+j+\frac{3}{2}h+\frac{3}{2}k+1$$

states. The maximization of $g(i,j,k,h)$ could be calculated by mathematics software such as Maple, but for the sake of completeness we include here a proof.[3] We have

$$g(i,j,k,h) = i(j+k+h+1)+j(k+h+1)+k(h+\frac{1}{2}k+\frac{3}{2})+\frac{1}{2}h^2+\frac{3}{2}h+1.$$

With $X = \frac{2k+3}{k+2}$, we get

$$g(i,j,k+X,h-1) = g(i,j,k,h)+(X-1)(i+j+k+h+\frac{1}{2}X+1) > g(i,j,k,h),$$

so under the restriction $h \ge m > k+1$, g is maximized by $h = k+2$. Now, $3k+3+i+j = n-1$, so $i = n-j-3k-4$, and after substitution in $g(i,j,k,h)$ we get

$$f(j,k) = -j^2+j(n-3k-4)-4k^2+k(2n-10)+3n-6.$$

1. Boundary $j = 0$. Then maximum is attained at $k = \frac{n-5}{8}$ and equals $\frac{7}{24}n^2+O(n)$.

[3] The proof was provided by one of the reviewers.

2. Boundary $k = 0$. Then maximum is attained at $j = \frac{n-4}{2}$ and equals $\frac{1}{4}n^2 + O(n)$.
3. Critical points: $\frac{\partial f}{\partial j} = -2j + n - 3k - 4$, $\frac{\partial f}{\partial k} = -8k - 5j + 2n - 15$. By solving $\frac{\partial f}{\partial j} = 0$ and $\frac{\partial f}{\partial k} = 0$, we get $k = \frac{1}{7}(n-8)$ and $j = \frac{2}{7}(n-1)$ (so that $i = \frac{2}{7}(n-1)$ and $h = \frac{1}{7}(n+6)$). This gives the maximal value of $\frac{2n^2+3n+2}{7}$ which is bounded from the above by $\frac{n^2+n+1}{3}$ for all $n \geq 1$.

This concludes the proof of the theorem. □

From Lemma 4.2, we already know that the upper bound of Theorem 5.4 can be reached by sets of equal length strings.

Corollary 5.5. *The upper bound for sets of equal length strings given by Theorem 5.4 is tight for all integers* $n \equiv 1 (\text{MOD } 3)$.

6. Conclusion

We have shown that the worst case state complexity of the permutation of a language over a binary alphabet recognized by an n-state chain DFA and, more generally, an n-state equal length DFA is $\frac{n^2+n+1}{3}$. We have given an upper bound $\frac{n^2-n+1}{2}$ for the state complexity of permutations of general finite languages over a binary alphabet, but the bound is not tight. The state complexity of permutation of finite languages depends on alphabet size and future work could consider finite languages over arbitrary alphabets.

Acknowledgments

We thank the anonymous referees for a careful reading of the paper and for useful suggestions. In particular, the non-optimality of the state complexity bound for general finite languages was pointed out by one of the referees and the referee also gave an explicit maximization proof for section 5.3 where we had done maximization using mathematics software Maple.

Cho and Han were supported by the Basic Science Research Program through NRF funded by MEST (2015R1D1A1A01060097), the Yonsei University Future-leading Research Initiative of 2016 and the IITP grant funded by the Korea government (MSIP) (R0124-16-0002). Goč, Palioudakis and Salomaa were supported by the Natural Sciences and Engineering Research Council of Canada Grant OGP0147224.

References

[1] Brzozowski, J. (2010). Quotient complexity of regular languages. *Journal of Automata, Languages, and Combinatorics*, *15*, 71–89.

[2] Brzozowski, J., Jirásková, G., Liu, B., Rajasekaran, A., & Szykula, M. (2016). On the state complexity of the shuffle of regular languages. In *Proceedings of DCFS 2016, Lect. Notes Comput. Sci. 9777* (pp. 73–86). Springer.

[3] Câmpeanu, C., Culik, K., Salomaa, K., & Yu, S. (2001). State complexity of basic operations on finite languages. In *Proceedings of WIA'99, Lect. Notes Comput. Sci. 2214* (pp. 60–70). Springer.

[4] Câmpeanu, C., Salomaa, K., & Yu, S. (2002). Tight lower bound for the state complexity of regular languages. *Journal of Automata, Languages and Combinatorics*, *7*, 303–301.

[5] Cui, B., Gao, Y., Kari, L., & Yu, S. (2012). State complexity of combined operations with two basic operations. *Theoretical Computer Science*, *429*, 82–102.

[6] Dassow, J., & Harbich, R. (2012). Descriptional complexity of union and star on context-free languages. *Journal of Automata, Languages and Combinatorics*, *17*, 123–143.

[7] Domaratzki, M., & Salomaa, K. (2008). Lower bounds for the transition complexity of nfas. *Journal of Computer and System Sciences*, *74*, 1116–1130.

[8] Eom, H.-S., Han, Y.-S., & Jiráskova, G. (2016). State complexity of basic operations on non-returning regular languages. *Fundamenta Informaticae*, *144*, 161–182.

[9] Gao, Y., Moreira, N., Reis, R., & Yu, S. (2015). A survey on operational state complexity. *Computer Science review (to appear), arXiv.org,* `http://arxiv.org/abs/1509.03254`*, 2015*, .

[10] Goldstine, J., Kappes, M., Kintala, C. M. R., Leung, H., Malcher, A., & Wotschke, D. (2002). Descriptional complexity of machines with limited resources. *Journal of Universal Computer Science*, *8*, 193–234.

[11] Gruber, H., & Holzer, M. (2007). On the average state and transition complexity of finite languages. *Theoretical Computer Science*, *387*, 155–166.

[12] Gruber, H., & Holzer, M. (2015). From finite automata to regular expressions and back — a survey of descriptional complexity. *International Journal of Foundations of Computer Science*, *26*, 1009–1040.

[13] Han, Y.-S., Salomaa, K., & Wood, D. (2009). Nondeterministic state complexity of basic operations on prefix-free regular languages. *Fundamenta Informaticae*, *90*, 93–106.

[14] Holzer, M., & Jacobi, S. (2012). Descriptional complexity of chop operations on unary and finite languages. *Journal of Automata, Languages and Combinatorics*, *17*, 165–183.

[15] Holzer, M., & König, B. (2004). On dfa and syntactic monoid size. *Theoretical Computer Science*, *327*, 319–347.

[16] Holzer, M., & Kutrib, M. (2003). Nondeterministic descriptional complexity of regular languages. *Int. J. Found. Comput. Sci.*, *14*, 1087–1102.

[17] Holzer, M., & Kutrib, M. (2003). Nondeterministic descriptional complexity of regular languages. *International Journal of Foundations of Computer Science*, *14*, 1087–1102.

[18] Holzer, M., & Kutrib, M. (2009). Nondeterministic finite automata — recent results on the descriptional and computational complexity. *International Journal of Foundations of Computer Science*, *20*, 563–580.

[19] Holzer, M., & Kutrib, M. (2011). Descriptional and computational complexity of finite automata - a survey. *Inf. Comput.*, *209*, 456–470.

[20] Krawetz, B., Lawrence, J., & Shallit, J. (2005). State complexity and the monoid of transformations of a finite set. *International Journal of Foundations of Computer Science*, *16*, 547–563.

[21] Kutrib, M., & Pighizzini, G. (2013). Recent trends in descriptional complexity of finite automata. *Bulletin of the EATCS*, *111*, 70–86.

[22] Lavado, G., Pighizzini, G., & Seki, S. (2013). Converting nondeterministic automata and context-free grammars into parikh equivalent one-way and two-way deterministic automata. *Information and Computation*, *228*, 1–15.

[23] Lavado, G., Pighizzini, G., & Seki, S. (2014). Operational state complexity under parikh equivalence. In *Proceedings of DCFC'14, Lect. Notes Comput. Sci. 8614* (pp. 294–305).

[24] Maslov, A. N. (1970). Estimates of the number of states of finite automata. *Dokl. Akad. Nauk SSSR*, *194*, 1266–1268.

[25] Palioudakis, A., Salomaa, K., & Akl, S. G. (2016). Operational state complexity of unary nfas with limited nondeterminism. *Theoretical Computer Science*, *610*, 108–120.

[26] Salomaa, A., Salomaa, K., & Yu, S. (2013). Undecidability of state complexity. *International Journal of Computer Mathematics*, *90*, 1310–1320.

[27] Salomaa, K., & Yu, S. (1997). NFA to dfa transformation for finite languages over arbitrary alphabets. *Journal of Automata, Languages, and Combinatorics*, *2*, 177–186.

[28] Shallit, J. (2009). *A Second Course in Formal Languages and Automata Theory*. Cambridge University Press.

[29] Yu, S. (1997). In G. Rozenberg, & A. Salomaa (Eds.), *Handbook of Formal Languages* chapter Regular Languages. (pp. 41–110). Springer volume 1.

[30] Yu, S. (2001). State complexity of regular languages. *Journal of Automata, Languages, and Combinatorics*, *6*, 221–234.

[31] Yu, S., Zhuang, Q., & Salomaa, K. (1994). The state complexities of some basic operations on regular languages. *Theoret. Comput. Sci.*, *125*, 315–328.